CPE301 – SPRING 2022

MIDTERM 2

Student Name: Ernesto Ibarra
Student #: 2001211571
Student Email: ibarre3@unlv.nevada.edu
Primary Github address: https://github.com/ErnestoIbarra333
Directory: https://github.com/ErnestoIbarra333/ErnestoIbarra

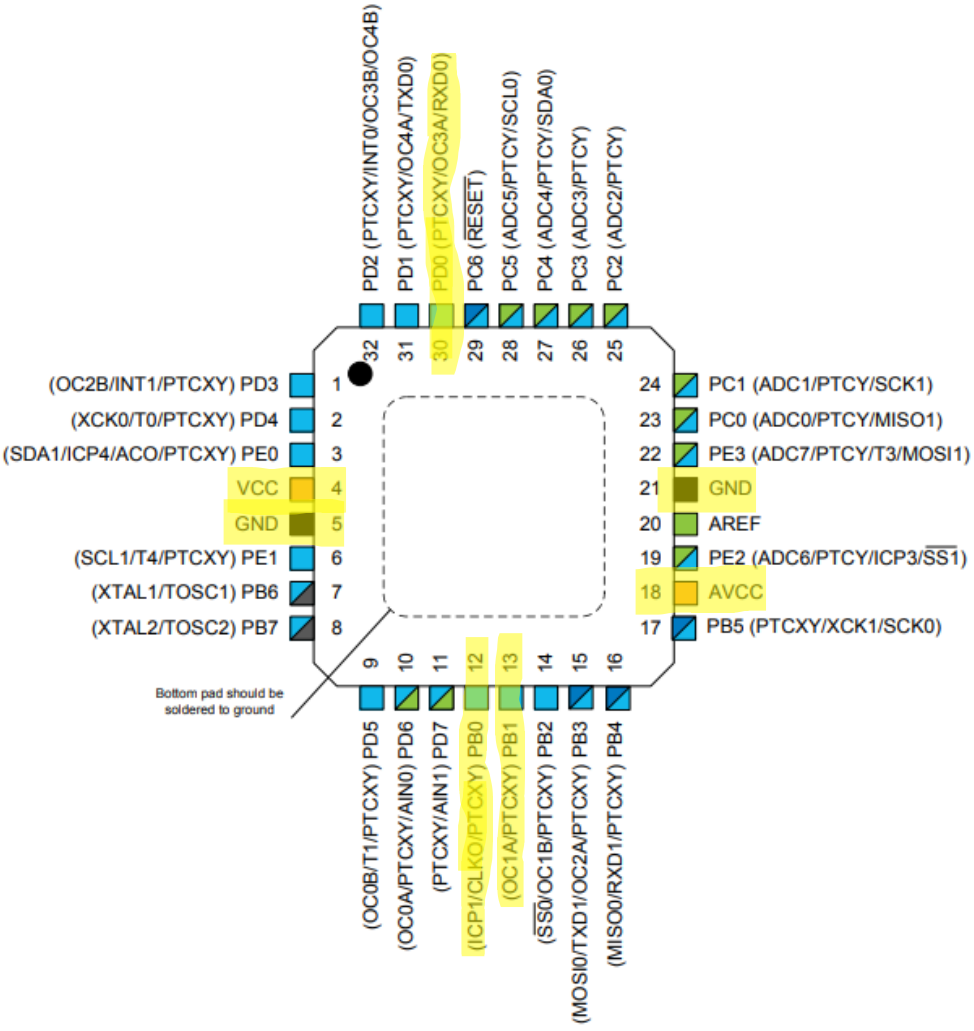
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmel Studio 7.0
- Assembler
- Simulator
- Debugger

Atmega328PB-Xmini

Multi-Function Shield
- Switches
- LEDs

Logic Analyzer



2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/2

Code for Atmel Studios, this controls the motor and the Ultrasonic Sensor

```
#define F_CPU 16000000UL
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <string.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define BAUDRATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)
#define  T_pin PB1 // pin for trigger
int TimerOF = 0; // overflow counter

//Simple Wait Function
void Wait()
{
    uint8_t i;
    for(i=0;i<8;i++)
    {
        _delay_loop_2(0);
    }

}

void main()
{
              char distance_string[10];
              long count2;
              double distance;
              double Angle = 80;
              DDRB = 0x02; // set PB1 and PB2 as outputs
              USART_init(); // initialize the USART
              TCCR1A = 0;  // start in normal mode
              TIMSK1 = (1 << TOIE1); // enable timer 1 overflow interrupt
              sei(); // enable interrupts

       TCCR3A |= (1<<COM3A1) | (1<<COM3B1) | (1<<WGM31);
    TCCR3B |= (1<<WGM33) | (1<<WGM32) | (1<<CS31) | (1<<CS30);
    ICR3 = 4999;

    DDRD|=(1<<PD0);   //PWM Pins as Out

    while(1)
    {
              double i = 115;
              while(i < 570) // here we start off the counter at 0 degrees which is 115
              {
                     OCR3A = i; //counter will go to 180 degrees which is 570
                     i = i + 6.5;
                     Wait(); // everytime the counter increments we take a snapshot of
the results of our Ultrasonic Sensor

                                   PORTB |= (1 << T_pin);
```

```
                                _delay_us(10); // add a quick trigger pulse for trigger
pin
                                PORTB &= (~(1 << T_pin));

                                TCNT1 = 0; // start timer at 0
                                TCCR1B = 0x41; // capture rising edge and with no pre
scalar
                                TIFR1 = 1<<ICF1; // clear ICP flag
                                TIFR1 = 1<<TOV1; // clear overflow flag

                                while ((TIFR1 & (1 << ICF1)) == 0); //We stay here
until rising edge
                                TCNT1 = 0; // start timer at 0
                                TCCR1B = 0x01; // capture falling edge instead now, no
pre scalar
                                TIFR1 = 1<<ICF1; // clear ICP flag
                                TIFR1 = 1<<TOV1; // clear overflow flag
                                TimerOF = 0; // clear our overflow timer

                                while ((TIFR1 & (1 << ICF1)) == 0); // we stay until
falling edge
                                count2 = ICR1 + (65535 * TimerOF); // receive value
from capture register
                                /* 8MHz Timer freq, sound speed = 343 m/s, calculation
mentioned in doc. */
                                distance = (double) count2 / (933);

                                 Angle = Angle + 2.5;
                                 dtostrf(Angle, 2, 0, distance_string);//turns distance
into a string
                                 strcat(distance_string, ","); // formatting
                                 USART_putstring(distance_string);    //prints to
terminal
                                 dtostrf(distance, 2, 0, distance_string);//turns
distance into a string
                                 strcat(distance_string, "."); // formatting

                                 USART_putstring(distance_string);    //prints to
terminal

              }

              double j = i;
              Angle = 180;
              while(j > 110) // Here we go backwards from 180 to 0 degrees
              {
                     OCR3A = j;
                     j = j - 6.5; // we subtract instead of add since we are going
backwards
                     Wait();
                                PORTB |= (1 << T_pin);
                                _delay_us(10); // add a quick trigger pulse for trigger
pin
                                PORTB &= (~(1 << T_pin));

                                TCNT1 = 0; // start timer at 0
                                TCCR1B = 0x41; // capture rising edge and with no pre
scalar
```

```
                                TIFR1 = 1<<ICF1; // clear ICP flag
                                TIFR1 = 1<<TOV1; // clear overflow flag

                                while ((TIFR1 & (1 << ICF1)) == 0); //We stay here
until rising edge
                                TCNT1 = 0; // start timer at 0
                                TCCR1B = 0x01; // capture falling edge instead now, no
pre scalar
                                TIFR1 = 1<<ICF1; // clear ICP flag
                                TIFR1 = 1<<TOV1; // clear overflow flag
                                TimerOF = 0; // clear our overflow timer

                                while ((TIFR1 & (1 << ICF1)) == 0); // we stay until
falling edge
                                count2 = ICR1 + (65535 * TimerOF); // receive value
from capture register
                                /* 8MHz Timer freq, sound speed = 343 m/s, calculation
mentioned in doc. */
                                distance = (double) count2 / (933);

                                 Angle = Angle - 2.5;
                                 dtostrf(Angle, 2, 0, distance_string);//turns distance
into a string
                                 strcat(distance_string, ","); // formatting
                                 USART_putstring(distance_string);    //prints to
terminal
                                 dtostrf(distance, 2, 0, distance_string);//turns
distance into a string
                                 strcat(distance_string, "."); // formatting

                                 USART_putstring(distance_string);    //prints to
terminal

                }
                Angle = 0;
    }
}

ISR(TIMER1_OVF_vect)
{
        TimerOF++; // here we increment our counter
}

void USART_init(void)
{

        UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);
        UBRR0L = (uint8_t)(BAUD_PRESCALLER);
        UCSR0B = (0<<RXEN0)|(1<<TXEN0);
        UCSR0C = (3<<UCSZ00);
}

void USART_send( unsigned char data)
{

        while(!(UCSR0A & (1<<UDRE0)));
        UDR0 = data;
```

```
}

void USART_putstring(char* StringPtr)
{

	while(*StringPtr != 0x00)
	{
		USART_send(*StringPtr);
		StringPtr++;
	}

}
```

Below is the code for the Processing Radar App which will display the Radar working using the Ultrasonic Sensor.

```
/*   Arduino Radar Project
 *
 *   Updated version. Fits any screen resolution!
 *   Just change the values in the size() function,
 *   with your screen resolution.
 *
 *  by Dejan Nedelkovski,
 *  www.HowToMechatronics.com
 *
 */

import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;

Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;

void setup() {

 size (1280, 720); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM11", 9600); // starts the serial communication
```

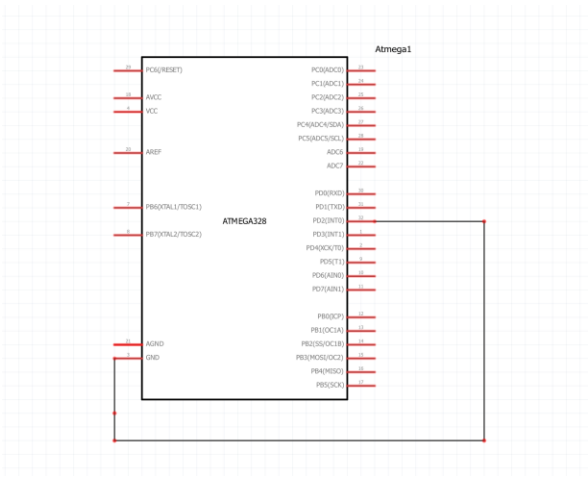
```
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually
it reads this: angle,distance.
 orcFont = loadFont("OCRAExtended-30.vlw");
}

void draw() {

  fill(98,245,31);
  textFont(orcFont);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable
"data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1"
  angle= data.substring(0, index1); // read the data from position "0" to position of the variable
index1 or thats the value of the angle the Arduino Board sent into the Serial Port
  distance= data.substring(index1+1, data.length()); // read the data from position "index1" to
the end of the data pr thats the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
```

```
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the
sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-
width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)));
// draws the line according to the angle
  popMatrix();
}

void drawText() { // draws the texts on the screen
```

```
  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Object: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("       " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-
width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-
width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-
width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-
width/2*sin(radians(120)));
```

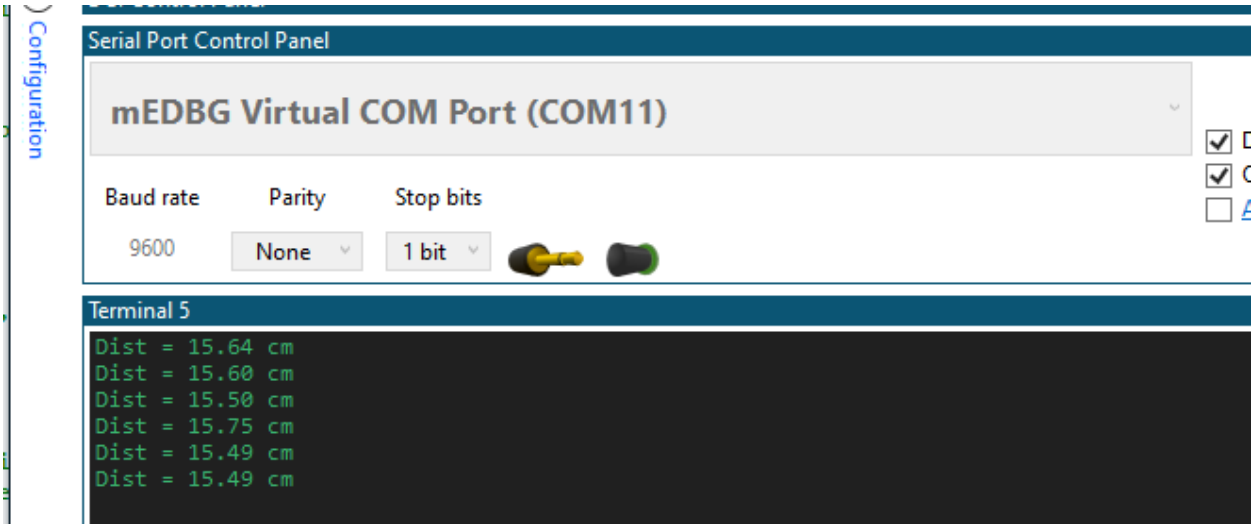
```
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-
width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```

3. SCHEMATICS

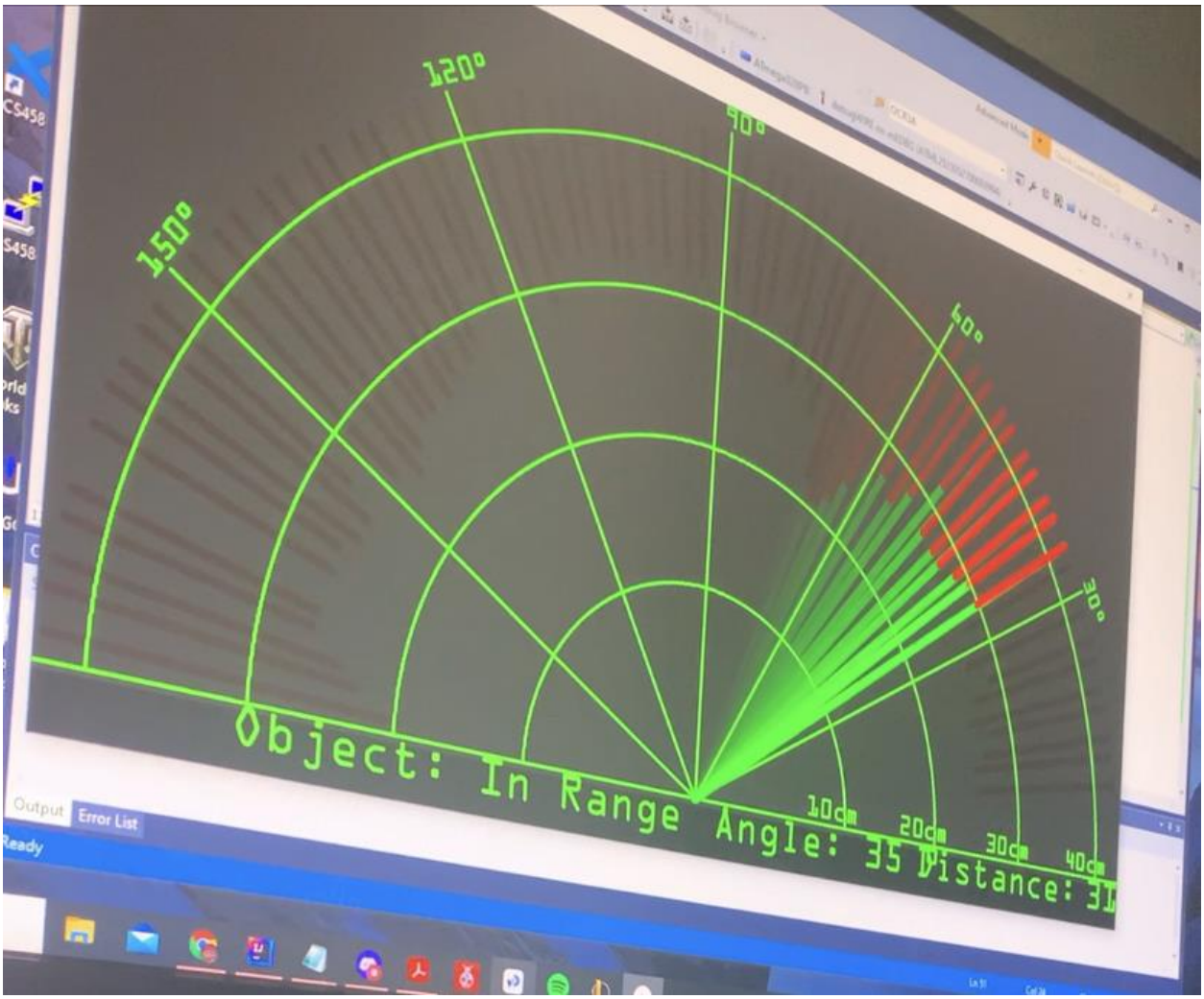


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

For part1 of this midterm 2 we needed to get our Ultrasonic Sensor working on top of our Servo Motor. I will show you the Ultrasonic Sensor glued on top my servo motor in the next section of this document. For now here is the output that shows my US working properly. Next is the part 2.

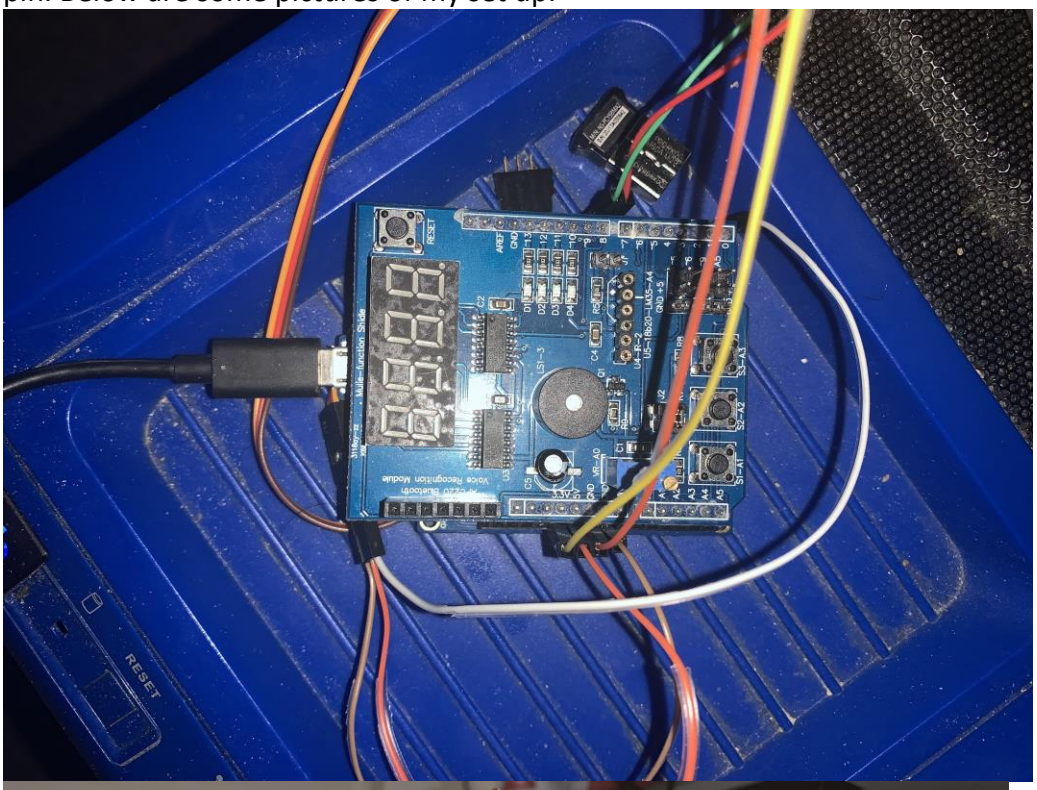


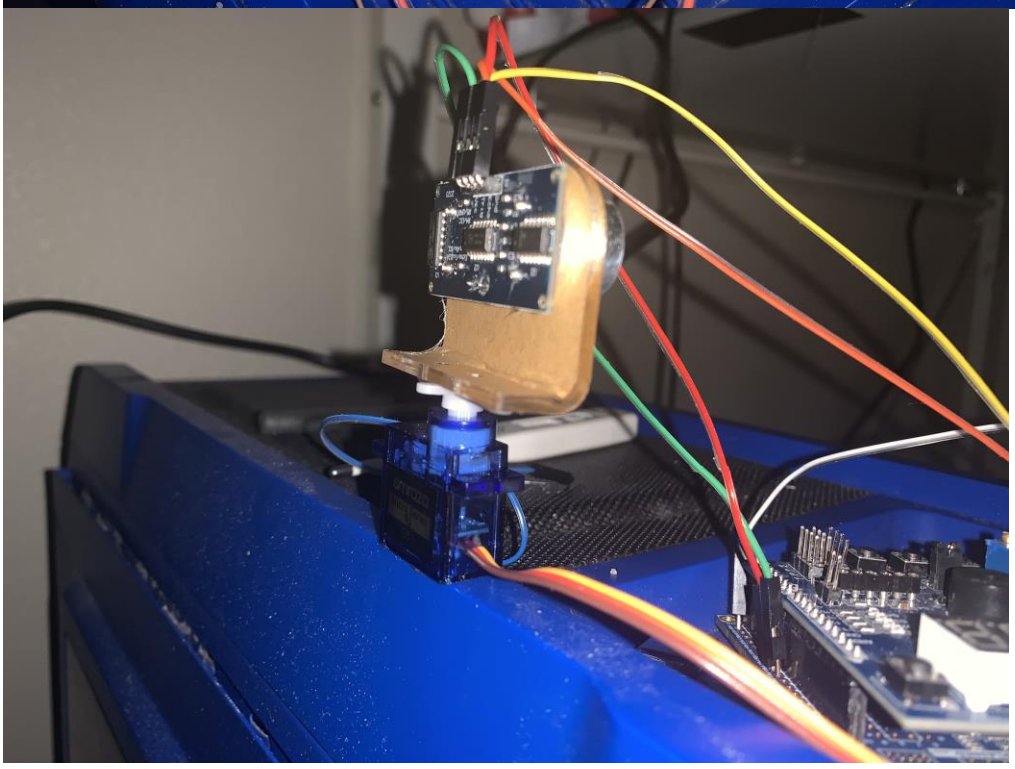
For Part2 of the midterm we needed to utilize a Radar application which will display our angle and distance. Below is a screenshot of what my display looked after connection my Ultrasonic Sensor on top of the Servo Motor and then connecting to the processing screen using the correct COM port.

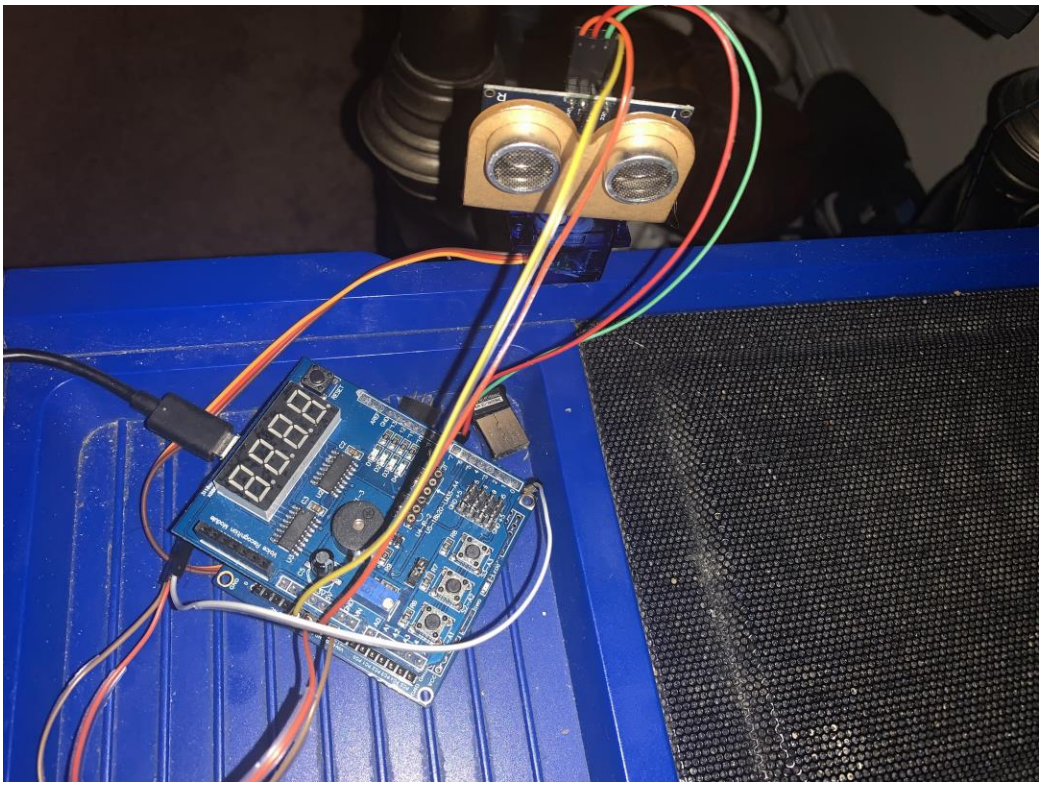


5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Here is a picture of my board set up for this midterm. You can see I am using VCC which is 5V and I'm also using the 3.3V to power the Ultra Sonic Sensor. I am also using PD0 to control my Servo Motor. For the ultrasonic sensor I am using PB0 for the Echo pin and PB1 for the trigger pin. Below are some pictures of my set up.







6. VIDEO LINKS OF EACH DEMO

Midterm Part 1: https://youtu.be/v0qYs1WCdaw
Midterm Part 2: https://youtu.be/NqL4MJjcuZk

7. GITHUB LINK OF THIS DA

https://github.com/ErnestoIbarra333/ErnestoIbarra/tree/main/Midterms/Midterm%202

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Ernesto Ibarra-Ayala